



# UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE  
United States Patent and Trademark Office  
Address: COMMISSIONER FOR PATENTS  
P.O. Box 1450  
Alexandria, Virginia 22313-1450  
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/551,246	04/17/2000	Alexander R. Krapf	C1064/7000(PJG/DPM)	9692
26161	7590	02/02/2004	EXAMINER	
FISH & RICHARDSON PC 225 FRANKLIN ST BOSTON, MA 02110			INGBERG, TODD D	
			ART UNIT	PAPER NUMBER
			2124	10
DATE MAILED: 02/02/2004				

Please find below and/or attached an Office communication concerning this application or proceeding.

**Office Action Summary**

Application No.

09/551,246

Applicant(s)

KRAPF ET AL.

Examiner

Todd Ingberg

Art Unit

2124

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

**Period for Reply**

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133).
- Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

**Status**

- 1) ☒ Responsive to communication(s) filed on 10 November 2003.
- 2a) ☐ This action is **FINAL**. 2b) ☒ This action is non-final.
- 3) ☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

**Disposition of Claims**

- 4) ☒ Claim(s) 1-48 and 75-145 is/are pending in the application.
- 4a) Of the above claim(s) 49-74 is/are withdrawn from consideration.
- 5) ☐ Claim(s) \_\_\_\_\_ is/are allowed.
- 6) ☐ Claim(s) 1-48 and 75-145 is/are rejected.
- 7) ☐ Claim(s) \_\_\_\_\_ is/are objected to.
- 8) ☐ Claim(s) \_\_\_\_\_ are subject to restriction and/or election requirement.

**Application Papers**

- 9) ☐ The specification is objected to by the Examiner.
- 10) ☒ The drawing(s) filed on 17 April 2000 is/are: a) ☐ accepted or b) ☒ objected to by the Examiner.
- Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).
- Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).
- 11) ☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

**Priority under 35 U.S.C. §§ 119 and 120**

- 12) ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
- a) ☐ All b) ☐ Some \* c) ☐ None of:
1. ☐ Certified copies of the priority documents have been received.
2. ☐ Certified copies of the priority documents have been received in Application No. \_\_\_\_\_.
3. ☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).
- \* See the attached detailed Office action for a list of the certified copies not received.
- 13) ☐ Acknowledgment is made of a claim for domestic priority under 35 U.S.C. § 119(e) (to a provisional application) since a specific reference was included in the first sentence of the specification or in an Application Data Sheet. 37 CFR 1.78.
- a) ☐ The translation of the foreign language provisional application has been received.
- 14) ☐ Acknowledgment is made of a claim for domestic priority under 35 U.S.C. §§ 120 and/or 121 since a specific reference was included in the first sentence of the specification or in an Application Data Sheet. 37 CFR 1.78.

**Attachment(s)**

- 1) ☒ Notice of References Cited (PTO-892)
- 2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
- 3) ☒ Information Disclosure Statement(s) (PTO-1449) Paper No(s) 5,6.
- 4) ☐ Interview Summary (PTO-413) Paper No(s). \_\_\_\_\_.
- 5) ☐ Notice of Informal Patent Application (PTO-152)
- 6) ☐ Other: \_\_\_\_\_.

Art Unit: 2124

## **DETAILED ACTION**

### **Pre-Restriction**

Claims 30, 31, 47 and 90 - 95 were amended on October 16, 2000.

Claims 105 - 145 were added on October 16, 2000.

### **Post-Restriction**

Group I - consisting of Claims 1 - 48, 75 - 145 have been examined.

Claims in Group II are considered cancelled (claims 49 - 74).

### ***Priority***

1. No official claim to priority has been made in the official record. Effective filing date is April 17, 2000.

### ***Information Disclosure Statement***

2. The information disclosure statements filed January 2, 2002 and September 6, 2002 have been considered. The reference Essential JNI JAVA Native Interface by Rob Gordon published 1998 is from the IDS.

### ***Drawings***

3. The drawings are deemed informal, formal drawings are required.

Art Unit: 2124

***Common Knowledge***

4. The following terms are common terms in the art and should have been known to one of ordinary skill in the art prior to the time of invention.

a. ***object*** – pages 271- 302

b. ***proxy*** – page 361

c. ***container*** – page 112

The definitions are provided in the “Dictionary of Object Technology” by Donald G. Firesmith et al. published September 22, 1995.

***Specification***

5. The use of the trademark JAVA has been noted in this application. It should be capitalized wherever it appears and be accompanied by the generic terminology.

Although the use of trademarks is permissible in patent applications, the proprietary nature of the marks should be respected and every effort made to prevent their use in any manner which might adversely affect their validity as trademarks.

***Claim Rejections - 35 USC § 112***

6. The following is a quotation of the second paragraph of 35 U.S.C. 112:

The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

Claims 11 – 48, 75 and 89 – 145 are rejected under 2173.05(m) PROLIX.

***Claim Rejections - 35 USC § 102***

7. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

A person shall be entitled to a patent unless –

Art Unit: 2124

(b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

8. Claims 1 - 8, 76 - 87, 102 - 104 are rejected under 35 U.S.C. 102(b) as being anticipated by Microsoft's commercial product Microsoft Foundation Class (MFC) library implemented in C++ as disclosed by Chapter 17 of the text book, "MFC with Visual C++ 5", by Mike Blaszcak, published April 19, 1997.

**NOTE:** This rejection is written to the level of a programmer. The MFC product is a programming library. The sections of the product selected for the rejection cover the use of proxies for collecting input from users and loading data in containers. Furthermore, the rejection covers the settings in MFC for features such as focus the ability to determine the focus on the screen and event handlers.

**Claim 1**

MFC anticipates in a computing system, a method for tracking user interaction with computer readable code, said method comprising: monitoring user interaction with computer readable code (MFC, pages 821 - 822, Event Notifications - event firing and the ability to detect such an event); recording information about the type of the user interaction (MFC, Control Proxy classes - Containers - pages 809 - 814) contain; if the user interaction relates to a point of focus in a first position in the computer readable code, then recording information about said first position; if the point of focus changes from said first position to a second position in the computer readable code (MFC, pages 821 - 822, Event Notifications - event firing and the ability to detect such an event), then determining if the content of the point of focus in said first position has changed (MFC, pages 821 - 822, Event Notifications - event firing and the ability to detect such an event); and if the content of the point of focus in said first position has changed (MFC, pages

Art Unit: 2124

821 - 822, Event Notifications - event firing and the ability to detect such an event), then recording information about the content of the point of focus in said first position (MFC, Control Proxy classes - Containers – pages 809 – 814).

**Claim 2**

The computer program product of claim 1, wherein the proxy component was generated by a transformation performed on the first component (MFC, page 820, Working with the Control – note the get and set ability).

**Claim 3**

The computer program product of claim 1, wherein the first semantic usability of the first component is defined at least by a first mutability, and wherein the second semantic usability of the proxy component is defined at least by a second mutability of the proxy component determined from at least the first mutability of the first component (MFC, C++ ability to call other methods – page 827 behave like an embedded object).

**Mutability** – is interpreted to be the form of polymorphism terms overloading.

Overloading is inherent in C++, in the overloading of methods based on what is passed.

**Claim 4**

The computer program product of claim 1, wherein the first semantic usability of the first component is defined at least by a first accessibility, and wherein the second semantic usability of the proxy component is defined at least by a second accessibility of the proxy component determined from at least the first accessibility of the first component.

Interpreted to be methods which are inherent in objects and C++ and supported in the rejection of claim 3.

Art Unit: 2124

**Claim 5**

The computer program product of claim 1, wherein the first semantic usability of the first component is defined at least by a first instantiability, and wherein the second semantic usability of the proxy component is defined at least by a second instantiability of the proxy component determined from at least the first instantiability of the first component.

Interpreted as instantiating proxy objects is inherent in the technology (MFC, page 781)

**Claim 6**

The computer program product of claim 1, wherein the first semantic usability of the first component is defined at least by a first inheritability, and wherein the second semantic usability of the proxy component is defined at least by a second inheritability of the proxy component determined from at least the first inheritability of the first component.

Interpreted as inherent technology of object technology as supported in the rejection of claim 5.

**Claim 7**

The computer program product of claim 1, wherein the first semantic usability of the first component is defined at least by a first context usability, and wherein the second semantic usability of the proxy component is defined at least by a context usability of the proxy component determined from at least the first context usability of the first component.

As per claim 1.

**Claim 8**

The computer program product of claim 1, wherein the first semantic usability of the first component is defined at least by a first polymorphicability, and wherein the second semantic

Art Unit: 2124

usability of the proxy component is defined at least by a polymorphicability of the proxy component determined from at least the first polymorphicability of the first component.

As per claim 3.

**Claim 76**

A system for modeling a first component of a first functional domain, wherein the first component defines a first concept and includes one or more subcomponents, the system comprising: means for receiving the first component; and means for generating a first model of the first component, including: means for generating, for each subcomponent of the first component, a discrete element of the first model to represent the subcomponent; and

means for providing the first model with a property of relationship awareness such that, if a first discrete element or attribute of the first model is changed, the first model is operative to: determine if the first discrete element or attribute has one or more elements and attributes related to the first discrete element or attribute, if the first discrete element or attribute has one or more related elements and attributes, determine whether to change the one or more related elements, and if it is determined to change one or more related elements and attributes, change such one or more elements and attributes in accordance with the changed first discrete element or attribute.

As per claim 1.

**Claim 77**

A system for modeling a first component of a first functional domain, when the first component defines a first concept and includes one or more subcomponents, the system comprising: a model generator to receive the first component, to generate a first model including: for each

Art Unit: 2124

subcomponent of the first component, a discrete element of the first model representing the subcomponent; and one or more model concepts that provide the first model with a property of relationship awareness such that, if a first discrete element or attribute of the first model is changed, the first model is operative to: determine if the first discrete element or attribute has one or more elements and attributes related to the first discrete element or attribute, if the first discrete element or attribute has one or more related elements and attributes, determine whether to change the one or more related elements, and if it is determined to change one or more related elements and attributes, change such one or more elements and attributes in accordance with the changed first discrete element or attribute.

As per claim 1.

**Claim 78**

A computer program product for modeling a first component of a first functional domain, wherein the first component defines a first concept and includes one or more subcomponents, the system comprising: a computer-readable medium having computer-readable signals stored thereon, wherein the signals define a model generator to receive the first component, to generate a first model including: for each subcomponent of the first component, a discrete element of the first model representing the subcomponent; and one or more model concepts that provide the first model with a property of relationship awareness such that, if a first discrete element or attribute of the first model is changed, the first model is operative to: determine if the first discrete element or attribute has one or more elements and attributes related to the first discrete element or attribute, if the first discrete element or attribute has one or more related elements and attributes, determine whether to change the one or more related elements, and if it is determined

Art Unit: 2124

to change one or more related elements and attributes, change such one or more elements and attributes in accordance with the changed first discrete element or attribute.

As per claim 1.

**Claim 79**

A method of transforming a first component of a first domain to a proxy component of a second domain, wherein the first component defines a first concept, the method comprising acts of:

- (a) determining a type of the first component; and
- (b) transforming the first component into the proxy component in

accordance with the determined type, wherein the proxy component defines at least the concept defined by the first component. As per claim 1.

**Claim 80**

The method of claim 79, wherein the first component is in parsed form. As per claim 1 – page 775 – a DLL to be added at link or run time by definition of a DLL.

**Claim 81**

The method of claim 79, wherein the first component is a source component.

As per claim 1 – page 775.

**Claim 82**

The method of claim 79, wherein the first component is a compiled component.

As per claim 1 – page 775.

**Claim 83**

The method of claim 79, wherein act (b) includes:

- (i) generating a model from the first component; and

Art Unit: 2124

(ii) generating the proxy component from the model.

As per claim 1.

**Claim 84**

The method of claim 79, wherein the proxy component has a semantic usability in the second domain closely corresponding to the semantic usability of the first component in the first domain. As per claim 1.

**Claim 85**

The method of claim 84, wherein the proxy component includes one or more proxy support elements to allow an instance of the proxy component to be contextaware. As per claim 1.

**Claim 86**

The method of claim 79, wherein the first domain is a first programming language. As per claim 1.

**Claim 87**

The method of claim 86, wherein the first programming language is a high-level programming language. As per claim 1

**Claim 102**

A system for transforming a first component of a first domain to a proxy component of a second domain, wherein the first component defines a first concept, the system comprising: means for determining a type of the first component; and means for transforming the first component into the proxy component in accordance with the determined type, wherein the proxy component defines at least the concept defined by the first component. As per claim 1.

Art Unit: 2124

**Claim 103**

A system for transforming a first component of a first domain to a proxy component of a second domain, wherein the first component defines a first concept, the system comprising: a component transformer to receive as input the first component, to determine a type of the first component, to transform the first component into the proxy component in accordance with the determined type, and to output the proxy component, wherein the proxy component defines at least the concept defined by the first component. As per claim 1.

**Claim 104**

A computer program product for transforming a first component of a first domain to a proxy component of a second domain, wherein the first component defines a first concept, the computer program product comprising:  
  
a computer-readable medium having computer-readable signals thereon, wherein the signals define a component transformer to receive as input the first component, to determine a type of the first component, and to transform the first component into the proxy component in accordance with the determined type, wherein the proxy component defines at least the concept defined by the first component. As per claim 1.

***Claim Rejections - 35 USC § 103***

9. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negated by the manner in which the invention was made.

Art Unit: 2124

10. Claims 9 – 48, 75 and 88 – 101 and 105 - 145 rejected under 35 U.S.C. 103(a) as being unpatentable over MFC as per above in view of Essential JNI JAVA Native Interface by Rob Gordon published 1998.

**Claim 9**

MFC teaches the computer program product of claim 1, wherein and the second functional domain is the C++ programming language as per claim 1, but MFC does not tech the first functional domain is the Java programming language. It is JNI who teaches the ability to communication between C++ and JAVA (v.v.) (JNI, pages 249 and page 270 and 278).

Therefore it would have been obvious to one of ordinary skill in the art at the time of invention to combine MFC and JNI , because the ability to share container information from user input makes the language of implementation transparent to the user.

**Claim 10**

The computer program product of claim 9, wherein the first component is a first Java component and the proxy component is a C++ proxy component, and wherein the C++ proxy component is aware of a context in which it is defined. As per claim 9.

**Claim 11**

The computer program product of claim 9, wherein the first component is a first Java component and the proxy component is a C++ proxy component, and wherein the C++ proxy component includes one or more proxy support elements. As per claim 9

**Claim 12**

The computer program product of claim 11, wherein the C++ proxy component is a C++ proxy class. As per claim 1.

Art Unit: 2124

**Claim 13**

The computer program product of claim 12, wherein one or more proxy support elements of the C++ proxy class allow an instance of the C++ proxy class to be context-aware. . As per claim 1.

**Claim 14**

The computer program product of claim 13, wherein one or more of the proxy support elements that allow an instance of the C++ proxy class to be contextaware are context constructors. As per claim 1.

**Claim 15**

The computer program product of claim 13, wherein one of the proxy support elements allows an instance of the C++ proxy class to be aware of being a C++ proxy instance field. (JNI, page 270, not isstatic)

**Claim 16**

The computer program product of claim 13, wherein one of the proxy support elements allows an instance of the C++ proxy class to be aware of being a C++ proxy static field. (JNI, page 270, isstatic)

**Claim 17**

The computer program product of claim 13, wherein one of the proxy support elements allows an instance of the C++ proxy class to be aware of being a C++ proxy array element. As per claim 1.

**Claim 18**

The computer program product of claim 13, wherein one of the proxy support elements allows an instance of the C++ proxy class to be aware of being a C++ proxy stand-alone object.

As per claim 16 – definition of a static object.

**Claim 19**

The computer program product of claim 13, wherein the C++ proxy class includes a proxy layer, and awareness of the contexts is required by the proxy layer. As per claim 1.

**Claim 20**

The computer program product of claim 19, wherein the proxy layer is coded using the Java Native Interface. As per claim 9.

**Claim 21**

The computer program product of claim 9, wherein the first component is a Java package and the proxy component is a C++ namespace.

As per claim 9 – the ability to call methods across the two languages and know the methods of other objects requires entries in namespace (inherent support for the mentioned functionality).

**Claim 22**

The computer program product of claim 9, wherein the first component is a Java class and the proxy component is either a C++ proxy class or a C++ proxy struct. As per claim 9.

**Claim 23**

The computer program product of claim 22, wherein the proxy component includes one or more proxy support elements. As per claim 1.

**Claim 24**

The computer program product of claim 23, wherein one or more of the proxy support elements allow an instance of the C++ proxy class to be context-aware. As per claim 1.

Art Unit: 2124

**Claim 25**

The computer program product of claim 22, wherein the Java class is declared abstract, and the C++ proxy component is instantiable. As per claim 9.

**Claim 26**

The computer program product of claim 9, wherein the first component is a Java array type and the proxy component is either a C++ proxy class or a C++ proxy struct. As per claim 9.

**Claim 27**

The computer program product of claim 26, wherein the proxy component includes one or more proxy support elements. As per claim 1.

**Claim 28**

The computer program product of claim 27, wherein one or more of the proxy support elements allow an instance of the proxy component to be context-aware. As per claim 1.

**Claim 29**

The computer program product of claim 26, wherein the proxy component includes a length field corresponding to a length attribute of the Java array. As per claim 9.

**Claim 30**

The computer program product of claim 26, wherein the Java array has an element type and the proxy component has an element type corresponding to the Java array's element type, and wherein the proxy component includes one or more array subscript operators that return a context-aware instance of the proxy component's type. As per claims 1 and 9.

Art Unit: 2124

**Claim 31**

The computer program product of claim 26, wherein the Java array has a primitive element type, and the proxy component has a primitive element type corresponding to the Java array's element type and includes one or more array subscript operators that return a primitive value. As per claim 9.

**Claim 32**

The computer program product of claim 9, wherein the first component is a Java interface and the proxy component is either a C++ proxy class or a C++ proxy struct. As per claim 9.

**Claim 33**

The computer program product of claim 32, wherein the first component includes one or more proxy support elements. As per claim 1.

**Claim 34**

The computer program product of claim 33, wherein one or more of the proxy support elements allow an instance of the proxy component to be context-aware. As per claim 1.

**Claim 35**

The computer program product of claim 32, wherein the proxy component is instantiable. As per claim 1.

**Claim 36**

The computer program product of claim 9, wherein the first component is a Java method and the proxy component is a C++ proxy method. As per claim 9.

Art Unit: 2124

**Claim 37**

The computer program product of claim 36, wherein the C++ proxy method has a constness determined from a mutability of the Java method. As per claim 9.

**Claim 38**

The computer program product of claim 36, wherein the C++ proxy method declares a return type that has a mutability determined from a mutability of a return type declared by the Java method. As per claim 9.

**Claim 39**

The computer program product of claim 36, wherein the C++ proxy method passes one or more arguments, each argument having a mutability determined from a mutability of a corresponding argument passed by the Java method. As per claim 9.

**Claim 40**

The computer program product of claim 36, wherein the C++ proxy method is defined to throw an exception based on the Java method being defined to throw an exception. As per claim 9 – ability to perform messaging in Object technology to throw an exception).

**Claim 41**

The computer program product of claim 36, wherein the C++ proxy method is defined not to throw an exception based on the Java method being defined to throw an exception. As per claim

Art Unit: 2124

**Claim 42**

The computer program product of claim 36, wherein whether the C++ proxy method is declared virtual or not declared virtual is determined from one or more of the following aspects of the Java method: polymorphicability, mutability, and instantiability. As per claim 9.

**Claim 43**

The computer program product of claim 9, wherein the first component is a Java field and the proxy component is a C++ proxy field. As per claim 9.

**Claim 44**

The computer program product of claim 43, wherein the C++ proxy field is declared to be of type C++ proxy class, the C++ proxy class including one or more proxy support elements that allow an instance of the C++ proxy class to be contextaware, such that an instance of the C++ proxy field is context-aware. As per claims 1 and 9.

**Claim 45**

The computer program product of claim 43, wherein the Java field is of primitive type and the C++ proxy field is declared to be of type C++ primitive proxy class. As per claim 9.

**Claim 46**

The computer program product of claim 45, wherein the C++ primitive proxy class includes one or more proxy support elements that allow an instance of the C++ primitive proxy class to be context-aware, such that an instance of the C++ proxy field is context-aware. As per claim 1.

**Claim 47**

The computer program product of claim 45, wherein one or more proxy support elements of the C++ primitive proxy class is an overloaded operator that permits instances the C++ primitive

Art Unit: 2124

proxy class 52 to be used on the left-hand side of one or more syntactical productions. As per claim 9.

**Claim 48**

The computer program product of claim 43, wherein the C++ proxy field has a mutability determined from a mutability of the Java field. As per claim 9.

**Claim 75**

The method of claim 9, wherein the second domain is a functional domain. As per claim 1.

**Claim 88**

The method of claim 87, wherein the first programming language is Java. As per claim 9.

**Claim 89**

The method of claim 88, wherein the second domain is C++. As per claim 9.

**Claim 90**

The method of claim 89, wherein: act (a) includes determining that the type of the first component is a Java class, and act (b) includes transforming the Java class into a C++ proxy class. As per claim 9.

**Claim 91**

The method of claim 90, wherein act (b) includes: generating, within the C++ proxy class, one or more proxy support elements. As per claim 9.

**Claim 92**

The method of claim 89, wherein: act (a) includes determining that the type of the first component is a Java array, and act (b) includes transforming the Java array into a C+ proxy class. As per claim 9.

**Claim 93**

The method of claim 92, wherein act (b) includes: generating, within the C++ proxy class, one or more proxy support elements. As per claim 9.

**Claim 94**

The method of claim 89, wherein: act (a) includes determining that the type of the first component is a Java method, and act (b) includes transforming the Java method into a C++ proxy method. As per claim 9.

**Claim 95**

The method of claim 89, wherein: act (a) includes determining that the type of the first component is a Java field, and act (b) includes transforming the Java field into a C++ proxy field. As per claim 9.

**Claim 96**

The method of claim 79, wherein the first semantic usability of the first component is defined at least by a first mutability, and act (b) includes:

(i) determining a second mutability of the proxy component from at least the first mutability of the first component, wherein the second semantic usability of the proxy component is defined at least by the second mutability.

As per claim 9.

**Claim 97**

The method of claim 79, wherein the first semantic usability of the first component is defined at least by a first accessibility, and act (b) includes:

Art Unit: 2124

(i) determining a second accessibility of the proxy component from at least the first accessibility of the first component, wherein the second semantic usability of the proxy component is defined at least by the second accessibility.

As per claim 9.

**Claim 98**

The method of claim 79, wherein the first semantic usability of the first component is defined at least by a first instantiability, and act (b) includes:

(i) determining a second instantiability of the proxy component from at least the first instantiability of the first component, wherein the second semantic usability of the proxy component is defined at least by the second instantiability.

As per claim 9.

**Claim 99**

The method of claim 79, wherein the first semantic usability of the first component is defined at least by a first inheritability, and act (b) includes:

(i) determining a second inheritability of the proxy component from at least the first inheritability of the first component, wherein the second semantic usability of the proxy component is defined at least by the second instantiability.

As per claim 9.

**Claim 100**

The method of claim 79, wherein the first semantic usability of the first component is defined at least by a first context usability, and act (b) includes:

Art Unit: 2124

(i) determining a second context usability of the proxy component from at least the first context usability of the first component, wherein the second semantic usability of the proxy component is defined at least by the second context usability.

As per claim 9.

**Claim 101**

The method of claim 79, wherein the first semantic usability of the first component is defined at least by a first polymorphicability, and act (b) includes:

(i) determining a second polymorphicability of the proxy component from at least the first polymorphicability of the first component, wherein the second semantic usability of the proxy component is defined at least by the second polymorphicability.

As per claim 9.

**Claim 105**

The computer program product of claim 12, wherein at least one of the proxy support elements of the C++ proxy class allows usage of null in C++ corresponding to usage of a null Java object reference in Java.

As per claim 9.

**Claim 106**

The computer program product of claim 105, wherein the C++ proxy class includes a conversion constructor to create a stand-alone proxy instance of the C++ proxy class initialized to not refer to any Java instance.

As per claim 9.

Art Unit: 2124

**Claim 107**

The computer program product of claim 12, wherein at least one of the proxy support elements provides a semantic usability to the C++ proxy class that closely corresponds to the semantic usability of a Java cast expression corresponding to the Java component.

As per claim 9.

**Claim 108**

The computer program product of claim 107, wherein the C++ proxy class includes a static class method that provides the semantic usability to the C++ proxy class that closely corresponds to the semantic usability of a Java cast expression corresponding to the Java component.

As per claim 9.

**Claim 109**

The computer program product of claim 12, wherein at least one of the proxy support elements provides an ability to map an instance of the C++ proxy class to an object that represents the Java component.

As per claim 9.

**Claim 110**

The computer program product of claim 109, wherein the C++ proxy class includes a framework support method that provides the ability to map an instance of the C++ proxy class to an object that represents the Java component.

As per claim 9.

**Claim 111**

The method of claim 89, wherein act (b) includes:

Art Unit: 2124

(i) transforming the Java component into a C++ proxy class that includes one or more proxy support elements.

As per claim 9.

**Claim 112**

The method of claim 111, wherein one or more proxy support elements of the C++ proxy class allow an instance of the C++ proxy class to be context-aware.

As per claim 1.

**Claim 113**

The method of claim 112, wherein one of the proxy support elements allows an instance of the C++ proxy class to be aware of being a C++ proxy instance field.

As per claim 1.

**Claim 114**

The method of claim 112, wherein one of the proxy support elements allows an instance of the C++ proxy class to be aware of being a C++ proxy static field.

As per claim 1.

**Claim 115**

The method of claim 112, wherein one of the proxy support elements allows an instance of the C++ proxy class to be aware of being a C++ proxy array element.

As per claim 1.

**Claim 116**

The method of claim 112, wherein one of the proxy support elements allows an instance of the C++ proxy class to be aware of being a C++ proxy stand-alone object.

Art Unit: 2124

As per claim 9.

**Claim 117**

The method of claim 112, wherein act (b)(i) includes: generating a proxy layer, and wherein awareness of the contexts is required by the proxy layer.

As per claim 9.

**Claim 118**

The method of claim 117, wherein act (b)(i) includes: coding the proxy layer using the Java Native Interface.

As per claim 9.

**Claim 119**

The method of claim 111, wherein at least one of the proxy support elements of the C++ proxy class allows usage of null in C++ corresponding to usage of a null Java object reference in Java.

As per claim 9.

**Claim 120**

The method of claim 119, wherein act (b)(i) includes: generating a conversion constructor within the C++ proxy class, the conversion constructor for creating a stand-alone proxy instance of the C++ proxy class initialized to not refer to any Java instance.

As per claim 9.

**Claim 121**

The method of claim 111, wherein at least one of the proxy support elements provides a semantic usability to the C++ proxy class that closely corresponds to the semantic usability of a Java cast expression corresponding to the Java component.

Art Unit: 2124

As per claim 9.

**Claim 122**

The method of claim 121, wherein act (b)(i) includes: generating a static class method ,within the C++ proxy class, the static class method providing the semantic usability to the C++ proxy class that closely corresponds to the semantic usability of a Java cast expression corresponding to the Java component.

As per claim 9.

**Claim 123**

The method of claim 111, wherein at least one of the proxy support elements provides an ability to map an instance of the C++ proxy class to an object that represents the Java component.

As per claim 9.

**Claim 124**

The method of claim 123, wherein act (b)(i) includes: generating a framework support method within the C++ proxy class, the framework support method providing the ability to map an instance of the C++ proxy class to an object that represents the Java component.

As per claim 9.

**Claim 125**

The method of claim 91, wherein one or more of the proxy support elements allow an instance of the C++ proxy class to be context-aware.

As per claim 1.

**Claim 126**

The method of claim 90, wherein the Java class is declared abstract, and act (b) includes:  
defining the C++ proxy component to be instantiable.

As per claim 9.

**Claim 127**

The method of claim 93, wherein one or more of the proxy support elements  
allow an instance of the proxy component to be context-aware.

As per claim 1.

**Claim 128**

The method of claim 92, wherein the Java array has a length attribute, and act (b) includes:  
defining the proxy class to include a length field corresponding to the length attribute of the Java  
array. As per claim 9.

**Claim 129**

The method of claim 92, wherein the Java array has an element type, and act (b) includes:  
defining the C++ proxy class to have an element type corresponding to the element type of the  
Java array; and generating, within the C++ proxy class, one or more array subscript operators  
that return a context-aware instance of the proxy class's type. As per claim 9.

**Claim 130**

The method of claim 92, wherein the Java array has a primitive element type, and act (b)  
includes: defining the C++ proxy class to have a primitive element type corresponding to the  
Java array primitive element type; and generating one or more array subscript operators that  
return a primitive value. As per claim 9.

Art Unit: 2124

**Claim 131**

The method of claim 94, wherein act (b) includes: defining the C++ proxy method to have a constness based on a mutability of the Java method. As per claim 9.

**Claim 132**

The method of claim 94, wherein act (b) includes: defining the C++ proxy method to declare a return type that has a mutability based on a mutability of a return type declared by the Java method. As per claim 9.

**Claim 133**

The method of claim 94, wherein act (b) includes: defining the C++ proxy method to pass one or more arguments; and defining each argument to have a mutability based on a mutability of a corresponding argument passed by the Java method. As per claim 9.

**Claim 134**

The method of claim 94, wherein act (b) includes: defining the C++ proxy method to throw an exception based on the Java method being defined to throw an exception. As per claim 9.

**Claim 135**

The method of claim 94, wherein act (b) includes: defining the C++ proxy method not to throw an exception based on the Java method being defined to throw an exception. As per claim 9.

**Claim 136**

The method of claim 94, wherein act (b) includes: determining whether the C++ proxy method is declared virtual or not declared virtual from one or more of the following aspects of the Java method: polymorphicability, mutability, and instantiability; and defining the C++ proxy method to be declared virtual or not declared virtual based on the determination. As per claim 9.

Art Unit: 2124

**Claim 137**

The method of claim 95, wherein act (b) includes: declaring the C++ proxy field to be of type C++ proxy class; and generating, within the C++ proxy class, one or more proxy support elements that allow an instance of the C++ proxy class to be context-aware, such that an instance of the C++ proxy field is-context-aware. As per claim 9.

**Claim 138**

The method of claim 95, wherein the Java field is of primitive type, act (b) including: defining the C++ proxy field to be declared of type C++ primitive proxy class. As per claim 9.

**Claim 139**

The method of claim 138, wherein act (b) further includes: declaring the C++ proxy field to be of type C++ proxy class; and generating, within the C++ proxy class, one or more proxy support elements that allow an instance of the C++ proxy class to be context-aware, such that an instance of the C++ proxy field is context-aware. As per claim 9.

**Claim 140**

The method of claim 138, wherein one or more of the proxy support elements of the C++ primitive proxy class is an overloaded operator that permits instances the C++ primitive proxy class 52 to be used on the left-hand side of one or more syntactical productions. As per claim 9.

**Claim 141**

The method of claim 95, wherein act (b) includes: determining a mutability of the Java field; and defining the C++ proxy field to have a mutability determined from the mutability of the Java field. As per claim 9.

Art Unit: 2124

**Claim 142**

The method of claim 89, wherein: act (a) includes determining that the type of the first component is a Java interface, and act (b) includes transforming the Java interface into a C++ proxy class. As per claim 9.

**Claim 143**

The method of claim 142, wherein act (b) includes: generating, within the C++ proxy class, one or more proxy support elements. As per claim 9.

**Claim 144**

The method of claim 143, wherein one or more of the proxy support elements allow an instance of the C++ proxy class to be context-aware. As per claim 9.

**Claim 145**

The method of claim 142, wherein act (b) includes: defining the C++ proxy class to be instantiable. As per claim 9.

***Correspondence Information***

Any inquiry concerning this communication or earlier communications from the examiner should be directed to **Todd Ingberg** whose telephone number is (703) 305-9775. The examiner can normally be reached during the following hours:

Monday	Tuesday	Wednesday	Thursday	Friday
6:15 – 1:30	6:15- 3:45	6:15 – 4:45	6:15-3:45	6:15-130

This schedule began December 1, 2003 and is subject to change.

Art Unit: 2124

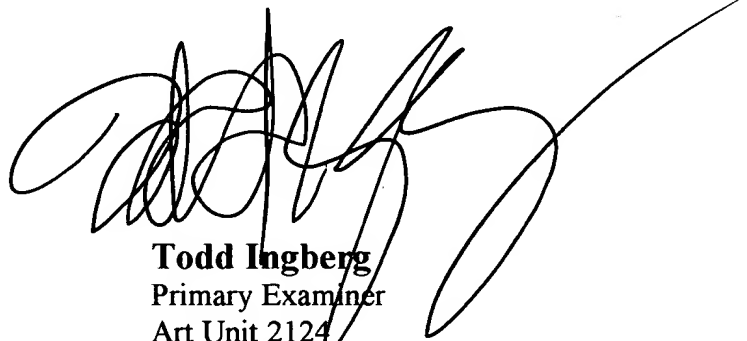
If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, **Kakali Chaki** can be reached on (703) 305-9662. Please, note that as of August 4, 2003 the **FAX number** changed for the organization where this application or proceeding is assigned is **(703) 872-9306**.

Also, be advised the United States Patent Office **new address** is

Post Office Box 1450

Alexandria, Virginia 22313-1450

Any inquiry of a general nature or relating to the status of this application or proceeding should be directed to the receptionist whose telephone number is (703) 305-9700.

A large, stylized handwritten signature in black ink, likely belonging to Todd Ingberg, is positioned above the printed name and title.

**Todd Ingberg**  
Primary Examiner  
Art Unit 2124  
January 25, 2004